

Do README ao AI Manifest: O Próximo Padrão dos Projetos de Software

SINOPSE

Definição da especificação do AI Manifest (.ai/manifest.md), um contrato declarativo de baixo acoplamento projetado para descrever a stack, arquitetura e limites operacionais de um repositório para agentes de IA.

Autor: Maicon Santos Webmaicon
Criado em: 10/06/2026

Assessora: Lunara LU-WANING 260710
Atualizado em: 19/06/2026

- AIManifestStandard
- DotAiDirectory
- ContextAsCode
- ContextEngineering
- AIAgentGovernance
- OperationalFirewall
- AgenticDevelopment
- ClaudeCode
- CursorAI
- GitHubCopilot
- WindsurfIDE
- AIIDEAdapters
- LogRotationChangelog
- ContextWindowOptimization
- PromptEngineering
- MultiAgentOrchestration

1. Resumo Executivo

Por décadas, os projetos de software foram estruturados sob a perspectiva de leitores humanos e ferramentas de automação tradicionais. Arquivos como `README.md`, `package.json`, `Dockerfile` e variáveis de ambiente `.env` definem como configurar, compilar, testar e distribuir um sistema.

Contudo, com a ascensão dos agentes autônomos de Inteligência Artificial integrados ao fluxo de desenvolvimento diário, surge uma nova necessidade metodológica: **os repositórios de código precisam se autodescrever de forma operacional para IAs.**

Embora o `README.md` continue indispensável para a integração de humanos, ele falha ao fornecer as restrições operacionais e os limites de atuação exigidos por um agente autônomo. O `AI Manifest` surge para preencher este gap: ele funciona como um contrato operacional explícito que dita arquitetura, políticas de permissão, stack tecnológica, padrões de qualidade e limites de segurança que governam as ações de agentes de IA dentro do ecossistema do projeto.

2. Premissas Fundamentais

IMPORTANT

A eficácia operacional de uma IA em um repositório é inversamente proporcional à sua necessidade de inferir ou improvisar.

A especificação do AI Manifest baseia-se em sete premissas:

1. **Natureza Declarativa:** A documentação para humanos é predominantemente narrativa; o contexto para agentes de IA deve ser declarativo, resumido e estruturado.
2. **Precedência da Arquitetura:** O agente não deve inferir ou reinventar padrões arquiteturais; ele deve aderir estritamente às decisões documentadas do repositório.
3. **Firewall Operacional:** As permissões de alteração, leitura e execução de comandos pelo agente devem ser explícitas, determinísticas e restritivas.
4. **Fonte Canônica da Verdade:** O repositório deve conter um único ponto de referência central de contexto, evitando a dispersão de instruções em múltiplos arquivos proprietários de fornecedores.
5. **Desacoplamento de IDEs:** adapters de ferramentas específicas (ex. Claude, Cursor, Copilot) devem ser derivados do contexto canônico, e não poluidores da raiz do repositório.
6. **Versionamento Nativo:** O manifesto operacional deve evoluir e ser versionado em conjunto com o código-fonte (`Context as Code`).
7. **Eficiência Cognitiva:** O contexto do projeto precisa ser modular, leve e de fácil ingestão para reduzir o consumo de tokens e evitar alucinações.

3. O Limite do README: O Gap do Desenvolvimento Assistido

O arquivo `README.md` foi projetado para responder ao "*como entender*" e "*como começar*" de um desenvolvedor humano. Ele explica o propósito macro do sistema e ensina a instalar dependências.

No entanto, um agente de IA executa ações de modificação direta no código. Para operar com segurança e precisão, ele necessita de respostas para perguntas que o `README.md` normalmente omite:

- Quais padrões de nomenclatura exatos devem ser aplicados para novas classes ou componentes?
- Quais diretórios e arquivos são estritamente sensíveis (firewall lógico)?
- Quais comandos de sistema são seguros para execução e quais exigem aprovação humana?
- Quais decisões arquiteturais passadas delimitam a implementação atual?
- Como fragmentar tarefas complexas sem violar a consistência do projeto?

Sem essas diretrizes declarativas, a IA opera sob um viés probabilístico genérico, o que leva a improvisações e, conseqüentemente, a regressões ou bugs de arquitetura.

4. A Proposta do AI Manifest

O **AI Manifest** centraliza e organiza todo o contexto operacional de IA em um diretório específico. A recomendação central de design é utilizar a pasta `.ai/` na raiz do projeto devido à sua semântica limpa e facilidade de varredura por assistentes.

```
.ai/
├─ manifest.md          # Contrato principal e indexador de regras
├─ core/                # Fonte canônica e regras de governança locais
│   ├── project.md      # Escopo de negócio e vocabulário do domínio
│   ├── stack.md        # Tecnologias oficiais e versões aceitas
│   ├── architecture.md # Padrões de software e organização de pastas
│   ├── permissions.md  # O Firewall Operacional (políticas de acesso)
│   ├── guidelines.md   # Padrões de código e convenções estilísticas
│   ├── testing.md      # Estratégia de testes e validação
│   ├── git.md          # Padrões de ramificação, commits e Pull Requests
│   ├── commands.md     # Lista de comandos autorizados e restritos
│   └─ risks.md         # Pontos de atenção e zonas críticas do sistema
├─ project/             # Base de conhecimento específica do projeto
│   └─ spec/            # Especificações técnicas (fonte de verdade para implementa
ção)
│   ├── context/        # Contextos técnicos – documentação viva por módulo (7 seções padrão)
│   │   ├── planning/   # Documentos estratégicos, backlog e revisões de fase
│   │   ├── memory/     # Snapshots de contexto da IA entre sessões
│   │   ├── product/    # Propostas de produto e exportações (PDFs)
│   │   ├── db/         # Diagramas de banco de dados e artefatos SQL
│   │   └─ scripts/     # Scripts de automação acoplados a este projeto
└─ shared/              # Capacidades e fluxos reutilizáveis
```

```

|   |— skills/           # Procedimentos passo a passo (capacidades da IA)
|   |— agents/          # Definições de papéis e personas de agentes
|   |— workflows/       # Rotinas e automações sequenciais
|   |— rules/           # Regras compartilhadas entre múltiplos projetos
|— output/              # Artefatos gerados intencionalmente pela IA
|— temp/                # Área transitória – não é fonte canônica
|— vendor/              # Dependências externas de contexto (somente leitura)
|— adapters/            # Configurações de tradução para IDEs específicas
|   |— claudes/          # Configurações para Claude Code / CLAUDE.md
|   |— codex/            # Configurações para Codex (OpenAI) / AGENTS.md
|   |— cursor/           # Configurações para Cursor (.cursorrules)
|   |— copilot/          # Configurações para GitHub Copilot
|   |— windsurf/         # Configurações para Windsurf
|— decisions/           # Registro de Decisões Arquiteturais (ADRs)
|   |— 0001-auth.md
|   |— 0002-billing.md
|— changelog.md         # Log ativo de alterações (< 500 linhas)
|— changelog/           # Governança e histórico rotacionado
|   |— README.md         # Regras de rotação e política de escrita
|   |— YYYY-MM-DD.md     # Arquivos históricos gerados por rotação

```

5. Detalhamento dos Arquivos Principais

1. `manifest.md`

É a regra zero. Funciona como ponto de entrada e contrato de inicialização da IA.

AI Manifest

Este repositório autoriza assistência por agentes de IA, sob a condição de aderência estrita às regras do diretório ``.ai/``.

Antes de propor ou modificar qualquer arquivo, o agente deve ler:

- ``.ai/core/permissions.md`` (prioridade máxima)
- ``.ai/core/project.md``
- ``.ai/core/architecture.md``

Em caso de ambiguidades de instrução, a política em ``.ai/core/permissions.md`` prevalece sobre qualquer outra regra.

2. `project.md`

Descreve o domínio de negócio de forma concisa para evitar que a IA trate a aplicação como um CRUD genérico.

Escopo do Projeto

Este sistema gerencia campanhas publicitárias conectando marcas a influenciadores digitais.

Termos de Domínio Fundamentais:

- * **Match Score:** Algoritmo que calcula afinidade entre marca e criador.
- * **Lockout Period:** Período obrigatório de não concorrência pós-campanha.
- * **Flag de Risco:** Marcação automática de potenciais conflitos éticos.

3. `stack.md`

Define o ecossistema tecnológico formal do projeto. Impede que a IA sugira ou injete dependências inadequadas.

Stack Tecnológica Oficial

Backend:

- * Laravel v11.x (PHP 8.3)
- * Pest (Framework de Testes)
- * PostgreSQL v16 (Banco de dados principal)

Frontend:

- * Livewire v3.x & Alpine.js
- * Tailwind CSS v3.x

4. `architecture.md`

Delimita as camadas do sistema e a separação de responsabilidades.

Diretrizes Arquiteturais

- * **Controllers:** Apenas orquestram chamadas; sem regras de negócio ou queries SQL.
- * **Services:** Concentram a lógica de negócio e regras do domínio.
- * **Form Requests:** Responsáveis únicos por validação de entrada de dados.
- * **Events/Listeners:** Utilizados para desacoplar ações secundárias (ex.: envio de e-mails).

5. `permissions.md` (O Firewall Operacional)

Garante a segurança da operação limitando o raio de ação das IAs. É o componente mais importante do manifesto.

```
# Políticas de Acesso e Escrita (Firewall)

## Zonas Livres (Sem necessidade de validação adicional)
* `app/Services/`
* `app/Http/Controllers/`
* `tests/`

## Zonas Restritas (Exigem confirmação explícita do desenvolvedor)
* Criação ou alteração de migrations de banco de dados.
* Fluxos críticos: Autenticação, Faturamento (Billing), e Permissões (ACL).

## Zonas Proibidas (Bloqueio total de alteração direta)
* Arquivos de configuração sensível (`env`, secrets, credenciais).
* Chaves criptográficas e certificados.
```

6. `guidelines.md`

Instrui sobre qualidade de código e convenções estilísticas recomendadas para o projeto.

```
# Diretrizes de Código

* Clean Code: Priorize nomes de variáveis e métodos descritivos antes de brevidade.
* Legibilidade: Código limpo e autodocumentado é preferível a comentários redundantes.
* Padrões: PHP em conformidade estrita com PSR-12.
```

6. O Subdiretório `project/` : Base de Conhecimento do Projeto

O diretório `core/` governa *como* a IA deve se comportar — suas regras, restrições e padrões operacionais. O diretório `project/` contém *o que* o projeto é — seu conhecimento de domínio, especificações técnicas e histórico decisório.

Essa separação é intencional: um agente deve aplicar `core/` como firewall antes de consultar `project/`. A mistura de conhecimento de domínio com regras de governança em um único diretório induz a IA a tratar restrições como sugestões e conteúdo como mandato.

Sub-pasta	Finalidade
<code>spec/</code>	Especificações técnicas — fonte de verdade para implementação
<code>context/</code>	Contextos técnicos — arquivos CONTEXT-*.md documentando cada módulo do sistema em 7 seções padronizadas (visão geral, fluxo principal, pontos críticos, tabelas, rotas, arquivos e métodos envolvidos)

<code>planning/</code>	Documentos estratégicos, backlog, revisões de fase e planos operacionais
<code>memory/</code>	Snapshots de contexto da IA entre sessões; gerados a partir das specs, não editados manualmente
<code>product/</code>	Propostas de produto e exportações (PDFs, apresentações)
<code>db/</code>	Diagramas de banco de dados e artefatos SQL
<code>scripts/</code>	Scripts de automação acoplados à estrutura de arquivos deste projeto (validadores, geradores)

Regra prática: Se o conteúdo descreve *o que o sistema faz*, pertence a `project/`. Se descreve *o que a IA pode ou não pode fazer*, pertence a `core/`.

`context/` : Documentação Técnica Viva

O subdiretório `context/` contém os **arquivos CONTEXT-*.md** — documentação estruturada e viva de cada módulo principal do sistema. Diferente de `spec/`, que abrigará especificações técnicas formais para implementações futuras, `context/` descreve o **estado atual e operacional** dos módulos existentes.

Cada arquivo CONTEXT-*.md segue obrigatoriamente **7 seções padronizadas**:

#	Seção	Conteúdo
1	Overview	Propósito do módulo, entidades principais, regras de negócio fundamentais
2	Main flow	Sequência de operações do início ao fim
3	Critical points	Invariantes, restrições e armadilhas que a IA não deve violar
4	Tables involved	Tabelas de banco de dados com colunas e relações relevantes
5	Routes involved	Endpoints HTTP com método, acesso e descrição
6	Files involved	Mapeamento de arquivos com papel de cada um
7	Methods involved	Métodos/funções com assinatura e descrição funcional

Por que 7 seções? Cada seção responde a uma pergunta diferente da IA: *o que é, como funciona, o que não quebrar, onde estão os dados, como acessar, quais arquivos tocar e quais métodos chamar*. Juntas, eliminam a necessidade de a IA inferir o contexto do módulo a partir do código-fonte bruto.

Quando criar um CONTEXT-*.md:

- Para cada módulo existente com lógica de negócio não trivial
- Antes de iniciar qualquer tarefa que afete o módulo
- Após mudanças arquiteturais significativas

Benefícios práticos:

- Reduz o consumo de tokens por sessão (contexto pré-processado vs. leitura de código bruto)
- Previne regressões ao documentar invariantes e pontos críticos explicitamente
- Permite que diferentes agentes (Claude, Cursor, Codex) recebam o mesmo contexto de referência
- Evolui junto com o código — versionado no repositório como *Context as Code*

`context/` **VS** `spec/` : `context/` documenta *como o sistema funciona hoje*; `spec/` descreve *como uma funcionalidade deve ser construída amanhã*.

`output/` e `temp/` : Artefatos Transitórios

- `output/` — Artefatos gerados intencionalmente pela IA (relatórios, exports, rascunhos de código). Podem ser versionados quando representam entregas formais.
- `temp/` — Área de staging transitória. Nunca deve ser usada como fonte canônica de verdade. Arquivos aqui são temporários e descartáveis; não os utilize quando existir correspondente em `project/`.

7. Modularidade e Reuso: `shared/`

Para acelerar tarefas comuns sem redundância de instrução no prompt do usuário, o manifesto organiza capacidades reutilizáveis no diretório `shared/`, dividido em quatro subdiretórios: `skills/`, `agents/`, `workflows/` e `rules/`.

`agents/` (Personas de Agentes)

Definem papéis, personas e escopos de responsabilidade para agentes especializados — por exemplo, um agente de revisão de código, um agente de documentação ou um agente de QA.

`rules/` (Regras Compartilhadas)

Regras de comportamento reutilizáveis entre múltiplos projetos da organização — convenções de nomenclatura, políticas de resposta, restrições globais de segurança que transcendem um único repositório.

`skills/` (Habilidades Procedimentais)

São guias passo a passo de como executar uma tarefa técnica repetitiva.

```
# Skill: Criar Endpoint de API
```

1. Analise o modelo de dados afetado.
2. Crie um Form Request específico para validação em ``app/Http/Requests``.
3. Implemente as regras de negócio em um Service exclusivo.
4. Exponha o endpoint via Controller, retornando uma Resource de resposta.
5. Escreva um teste de integração (Feature test) cobrindo caminhos felizes e de erro.

workflows/ (Fluxos de Processo)

Definem rituais operacionais do ciclo de desenvolvimento.

Workflow: Nova Feature

1. Leia o escopo da tarefa e consulte ``project.md``.
2. Identifique os módulos afetados e busque referências em ``architecture.md``.
3. Desenvolva a funcionalidade acompanhada de seus testes.
4. Execute a suite de testes locais.
5. Registre a mudança em ``changelog.md`` seguindo a semântica do projeto.

8. Resolução de Conflitos e Precedência

Em cenários complexos onde regras do time, da empresa ou da IA colidem, a árvore de precedência é executada de forma determinística de cima para baixo:

1. permissions.md (Regra de segurança local do repositório)
 - └─ 2. architecture.md (Padrão arquitetural do repositório)
 - └─ 3. decisions/ (Decisões arquiteturais históricas locais)
 - └─ 4. stack.md (Stack tecnológica local)
 - └─ 5. guidelines.md (Padrões de estilo locais)
 - └─ 6. Regras Corporativas (Diretrizes globais da empresa)
 - └─ 7. Preferências Pessoais (Configurações do desenvolvedor)

9. Abstração por IDEs: A Camada de Adapters

Provedores de IA utilizam arquivos de instrução proprietários na raiz do repositório (`CLAUDE.md`, `AGENTS.md`, `.cursorsrules`). O padrão resolve isso com dois níveis:

1. **Arquivo index na raiz** — arquivo mínimo exigido pelo agente; aponta para os arquivos canônicos em `.ai/`.
2. **Arquivo canônico no adapter** — conteúdo completo de contexto e convenções em `.ai/adapters/<ide>/`.

```

CLAUDE.md (raiz - index)      → .ai/adapters/claude/CLAUDE.md (conteúdo real)
AGENTS.md (raiz - index)     → .ai/adapters/codex/AGENTS.md (conteúdo real)
.cursorrules (raiz - index)  → .ai/adapters/cursor/          (conteúdo real)

```

Estrutura do arquivo index

O index na raiz deve ser mínimo e declarativo, apontando para os arquivos canônicos na ordem de precedência:

CLAUDE.md

Antes de iniciar qualquer tarefa, leia os seguintes arquivos nesta ordem:

1. ``.ai/manifest.md`` – contrato do projeto
2. ``.ai/adapters/claude/CLAUDE.md`` – contexto completo, regras e convenções
3. ``.ai/changelog.md`` – registro das alterações recentes

Em caso de conflito, a ordem acima define a precedência.

Mapeamento por IDE

Agente	Index (raiz)	Canônico (<code>.ai/adapters/`</code>)
Claude Code	<code>CLAUDE.md</code>	<code>.ai/adapters/claude/CLAUDE.md</code>
Codex (OpenAI)	<code>AGENTS.md</code>	<code>.ai/adapters/codex/AGENTS.md</code>
Cursor	<code>.cursorrules</code>	<code>.ai/adapters/cursor/</code>
GitHub Copilot	<code>.github/copilot-instructions.md</code>	<code>.ai/adapters/copilot/</code>
Windsurf	<code>.windsurf/rules/</code>	<code>.ai/adapters/windsurf/</code>

Desta forma, os desenvolvedores atualizam apenas a documentação canônica em

`.ai/adapters/<ide>/` e o arquivo index na raiz permanece estável e mínimo.

10. Governança do Changelog: Log Rotation

O arquivo `.ai/changelog.md` cresce continuamente — novas entradas são sempre inseridas no topo, empurrando o histórico para baixo. À medida que o projeto avança, o arquivo cresce indefinidamente. Inspirado no `logrotate` de servidores Linux, o padrão define uma política de rotação automática por volume de linhas.

Estrutura

```
.ai/
├─ changelog.md           ← log ativo (entradas recentes)
├─ changelog/
│   ├─ README.md         ← regras de governança e rotação
│   ├─ 2026-06-14.md     ← arquivo histórico (gerado por rotação)
│   └─ 2026-12-01.md     ← próximo arquivo histórico
```

Política de Rotação

Parâmetro	Valor
Gatilho	<code>changelog.md</code> atinge 500 linhas
Nome do arquivo histórico	<code>YYYY-MM-DD.md</code> — data em que a rotação foi executada
Destino	<code>.ai/changelog/</code>
Entradas preservadas no ativo	Últimas ~50 linhas

Fluxo de Rotação

```
# 1. Verificar contagem
wc -l .ai/changelog.md

# 2. Gerar nome do arquivo histórico
date +%Y-%m-%d # ex: 2026-06-14

# 3. Mover entradas antigas → .ai/changelog/2026-06-14.md
# 4. Adicionar referência ao final do changelog ativo:
# *Entradas anteriores: [changelog/2026-06-14.md](changelog/2026-06-14.md)*
```

Regras

- O arquivo histórico é **somente leitura** após gerado — nunca edite arquivos de rotação
- O nome do arquivo reflete a **data da rotação**, não a data das entradas
- Agentes de IA leem apenas `.ai/changelog.md`; arquivos históricos são para referência humana
- Regras completas de governança ficam em `.ai/changelog/README.md`

11. Análise de Riscos e Mitigações

- **Risco de Obsolescência:** Documentos de contexto desatualizados induzem a IA a introduzir padrões legados no código.

- **Mitigação:** Valide a consistência das regras no pipeline de Integração Contínua (CI), verificando se modificações arquiteturais foram acompanhadas de atualizações na pasta `.ai/`.
- **Excesso de Especificação:** A imposição de regras excessivamente rígidas limita a capacidade de resolução de problemas do agente de IA.
 - **Mitigação:** O manifesto deve focar em *limites e restrições*, não em microgerenciamento de algoritmos.
- **Injeção de Contexto Malicioso:** Pacotes em `vendor/` importados de fontes não confiáveis podem conter prompts maliciosos projetados para exfiltrar segredos.
 - **Mitigação:** Realize revisões de segurança e assine digitalmente dependências de contexto corporativo.
- **Dispersão de Conhecimento em Regras de Governança:** Quando specs, planejamento e memória de projeto ficam misturados com regras de governança em `core/`, a IA confunde restrição comportamental com conhecimento de domínio — tratando limitações como sugestões e conteúdo como mandato.
 - **Mitigação:** Use `core/` exclusivamente para regras de comportamento da IA e `project/` para todo o conhecimento do domínio.

12. Roadmap Conceitual de Adoção

```
graph TD
  F1[Fase 1: Convenção Manual] --> F2[Fase 2: Templates Estruturados]
  F2 --> F3[Fase 3: Geração de Adapters]
  F3 --> F4[Fase 4: Linter de Contexto]
  F4 --> F5[Fase 5: Integração em Runtime]
```

- **Fase 1 — Convenção Manual:** Implementação direta do diretório `.ai/` e seus arquivos fundamentais no repositório.
- **Fase 2 — Templates Estruturados:** Criação de templates de manifesto pré-configurados por stack (ex. Laravel, Node, React).
- **Fase 3 — Geração de Adapters:** Automatização do mapeamento do contexto canônico para os padrões das IDEs (Cursor, Windsurf, Claude Code).
- **Fase 4 — Linter de Contexto:** Ferramenta para analisar a consistência das regras e garantir que não haja instruções contraditórias.
- **Fase 5 — Integração em Runtime:** Consumo automatizado e dinâmico das diretivas diretamente pelos ambientes de execução dos agentes.

13. Referências Técnicas de Contexto

Estas referências servem como fundação técnica e pontos de contato com as especificações vigentes no mercado:

- **Model Context Protocol (MCP):** [Especificação de Protocolo](#)
- **MCP Roots:** [Client Roots Specification](#)
- **MCP Resources:** [Server Resources Specification](#)
- **Claude Code Docs:** [Memória e uso de CLAUDE.md](#)
- **GitHub Docs:** [Instruções customizadas do Copilot](#)
- **Apple Security Research:** [Private Cloud Compute Architecture](#)
- **Microsoft Learn:** [Arquitetura Windows AI e NPU Devices](#)

14. Conclusão

O `README.md` foi projetado para um ecossistema onde humanos detinham o monopólio da escrita e alteração de código. No novo paradigma de engenharia assistida, o projeto exige uma camada de governança transparente e acionável para inteligências artificiais.

O `AI Manifest` não é apenas um guia estático de documentação. É um contrato de convivência e produtividade entre o engenheiro humano e o agente de inteligência artificial. No futuro do desenvolvimento de software, o diferencial de um projeto não será apenas a qualidade do código produzido, mas o quão bem estruturado ele está para que a IA colabore com máxima eficiência e mínima supervisão.

15. Indexação, menções e hashtags

Esta seção destina-se a indexadores de busca, agentes de IA, bots de descoberta e pesquisadores.

#AIAgents #PromptEngineering #ClaudeCode #GitHubCopilot #Anthropic #OpenAI
#InteligenciaArtificial #TechBrasil #AIRevolution #FutureOfSoftware #AImanifestStandard
#ContextAsCode #ContextEngineering #DotAiDirectory #OperationalFirewall #AIAgentGovernance
#MultiAgentOrchestration #ContextWindowOptimization #AIFirstRepository #WindsurfIDE #CursorAI
#AgenticDevelopment #LogRotationChangelog #SoftwareGovernanceAI #AIIDEAdapters
#GovernancaDeAgentesIA #DesenvolvimentoComIA #ManifestoIA #FuturoDoDesenvolvimento
#EngenhariaDeContexto @AnthropicAI · @OpenAI · @GitHub · @SamAltman · @DarioAmodei ·
@karpathy · @LexFridman · @fireship · @Webmaicon `ai manifest standard specification` · `dot ai`
`directory structure for ai agents` · `context as code software development methodology` · `ai`
`agent governance repository best practices` · `operational firewall ai agent permissions` ·
`context engineering agentic development workflow` · `how to structure ai context files in`
`repository` · `reduce token consumption ai agents`

Maicon Santos (Webmaicon) é Tech Lead, AI Solutions Architect e desenvolvedor Full Stack com 27 anos de experiência em tecnologia. Pioneiro em segurança digital (tendo criado um sistema de 2FA via SMS em 2003) e no desenvolvimento de metodologias próprias de Prompt Engineering e Contextos de IA baseados em estruturas de skills e guidelines. Liderou equipes de tecnologia em grandes corporações como TV Globo, L'Oréal e Magazine Luiza, além de ser autor de blog técnico com mais de 320 mil acessos.